IN THE CLAIMS

Please amend the claims as follows.

1. (Currently Amended) A method of compiling code, comprising:

partitioning instructions in the code among a plurality of processors based on memory access latency associated with the instructions by partitioning memory access dependence chains into an upstream stage by assigning a first number of desired upstream nodes to the upstream stage, and assigning instructions in the code which the first number of desired upstream nodes are dependent to the upstream stage; and

partitioning the instructions among the plurality of processors by partitioning the memory access dependence chains into a downstream stage by assigning a last number of desired downstream nodes to the downstream stage, and assigning instructions in the code which are dependent on the last number of desired downstream nodes to the downstream stage.

Claims 2-4. (Canceled)

5. (Currently Amended) The method of Claim 41, wherein the number of desired upstream nodes is the length of the memory access dependence chain divided by a pipelining degree.

6. (Currently Amended) The method of Claim 41, further comprising:

generating a new number of desired upstream nodes if a computed weight of the upstream stage exceeds a predetermined value; and

assigning ~~a first~~the new number of desired upstream nodes to the upstream stage; and

assigning instructions in the code on which the ~~first~~ new number of desired upstream nodes are dependent to the upstream stage.

Claims 7-8 (Canceled)

9. (Currently Amended) The method of Claim ~~8~~1, wherein the number of desired downstream nodes is N*(d-1)/d, where N is a length of the memory access dependence chain, and d is a pipelining degree.

10. (Original) The method of Claim 1, further comprising identifying instruction dependence information.

11. (Original) The method of Claim 1, further comprising constructing a memory access dependence graph.

12. (Original) The method of Claim 1, further comprising:

constructing a memory access dependence graph; and

identifying a memory access dependence chain from the memory access dependence graph.

13. (Previously Presented) An article of manufacture comprising a non-transitory machine accessible medium including sequences of instructions, the sequences of instructions including instructions which when executed cause the machine to perform:

partitioning instructions in code into a plurality of pipeline stages to be executed in parallel by a plurality of processors based on memory access latency associated with the instructions.

14. (Previously Presented) The article of manufacture of Claim 13, further comprising instructions which when executed causes the machine to further perform constructing a memory access dependence graph.

15. (Currently Amended) The article of manufacture of Claim 13, wherein partitioning instructions comprises partitioning a memory access dependence chain into an ~~upstage~~ upstream stage by assigning a first number of desired upstream nodes to the upstream stage, and assigning instructions in the code on which the first number of desired upstream nodes are dependent to the upstream stage.

16. (Previously Presented) A code analysis unit implemented on a processor, comprising:

a dependence information unit to identify dependencies between instructions in code; and

a code partitioning unit to partition instructions in the code into a plurality of pipeline stages to be executed by a plurality of processors based on memory access latency associated with the instructions.

17. (Previously Presented) The apparatus of Claim 16, wherein the code partition unit comprises:

a length unit to determine a number of nodes from a memory access dependence chain to assign to an upstream stage;

an assignment unit to assign a first number of desired nodes to the upstream stage;

a close up unit to assign instructions in the code for which the first number of desired length of upstream nodes are dependent to the upstream stage; and

an evaluation unit to determine whether a computed weight of the upstream stage exceeds a predetermined value.

18. (Previously Presented) The apparatus of Claim 17, wherein the length unit determines a new number of desired length of upstream nodes in response to the evaluation unit determining that the computed weight of the upstream stage exceeds the predetermined value.

19. (Currently Amended) The apparatus of Claim ~~16~~17, wherein the length unit determines a number of nodes from the memory access dependence chain to assign to a downstream stage, the assignment unit assigns a first number of desired nodes to the downstream stage, the close up unit assigns instructions in the code on which are dependent on the first number of desired length of down stream nodes, and an evaluation unit to determine whether a computed weight of the downstream stage exceeds the predetermined value.

20. (Previously Presented) The apparatus of Claim 19, further comprising a balancing unit to assign remaining instructions to the upstream stage and the downstream stage in a manner that substantially balances computed weight.

21. (New) The apparatus of Claim 17, further comprising a code partition manager that receives instruction dependence information and the memory access dependence chains.

22. (New) The apparatus of Claim 16, wherein the dependence information unit generates a memory access dependence graph and memory access dependence chains from instruction dependence information.